Q: Why is a Raven like a Writing Desk? A: They're both Objects.

Keith Duddy
Queensland University of Technology
Brisbane, Australia
keith.duddy@qut.edu.au

Jörg Kiegeland
Queensland University of Technology
Brisbane, Australia
joerg.kiegeland@qut.edu.au

Scott Beazley
Project Services, Queensland Government
Brisbane, Australia
scott.beazley@projectservices.qld.gov.au

Jim R H Steel
University of Queensland
Brisbane, Australia
j.steel@uq.edu.au

ABSTRACT

The famous riddle by Lewis Carroll is not really intended to have an answer[1], but rather to reveal the incomprehension Alice has of the world of adults. This paper is about a team of model-driven software engineers' concepts of "object" coming up against the building design sector's concept of "object", and challenging our assumptions about the clean and clever solutions that we can create with object-oriented metamodels and model transformation and code generation tools. However, the story has a happy ending in which the application of our tools in combination with pragmatic choices of representations of building designs still produces an outcome that meets our users' needs and avoids lots of bespoke programming.

1. INTRODUCTION

Through the Sustainable Built Environment National Research Centre, QUT is engaged with Queensland Government and other partners in producing a candidate technology to fill the role of an Australian National Object Library. In this context "objects" are re-usable design components for CAD (Computer Aided Design) software. The project scope requires that the library be independent of any CAD-tool-specific format or dependency, but that designers must be able to access the library from within their CAD tools, and download or attach the relevant product information directly into their designs.

The Industry Foundation Classes (IFC) specification [3] is a metamodel of buildings which allows for the exchange of structured information among tools for Computer Aided Drafting (CAD) and Computer Aided Engineering (CAE). The IFC specification is designed by CAD/CAE vendors and design practitioners specifically for the purpose of moving designs between tools, which all use their own internal representations and file formats. Initially we assumed that this would be the only model type needed to store a set of representing products available from manufacturers.

We initially chose to use an EMF [12] implementation of IFC as a tool-independent representation for our online library of building product descriptions. In their EMF form, IFC models appeared to be amenable to transformation into CAD-specific formats with the tools which our team had to hand, namely: an EMF-based IFC resource implementation that inputs/outputs standards-compliant IFC files [11], a repository generation tool which creates a Web services layer over a scalable model database [6, 5], and a QVT-like transformation engine [8, 10]. The working description we had of "products lines" was that they were a standard set of properties defined by a template, along with an optional geometric shape. This proved to have nuances that defied easy implementation in the strictly object-oriented EMF environment. In addition, the ability to provide a set of product templates by leveraging published international standards in a variety of formats was necessary, as the resources to specify required properties for the thousands of product types used in the Australian construction sector were not available.

This paper discusses the resolution of tensions between the object-oriented notions of type and instance, and the requirement for representing both product templates, which define the "type" of a product, and fully-described product lines, which "instantiate" a template, as EMF objects. The naive assumption that the product template/product description relationship would mirror the EMF Class/EObject *instantiation* relationship was rapidly dispelled as soon as we starting analysing the requirements of our largest stake-

[1]In an introduction to a later edition of *Alice in Wonderland*, Lewis Carroll suggested the following answer: "Because it can produce a few notes, tho they are very flat; and it is nevar put with the wrong end in front!" Note the use of "nevar" instead of "never", which is "raven" spelled backwards.

holder, the Queensland State Government's Project Services agency. We discovered that supporting an building design practice with information rich design objects requires parts of the information to be added incrementally, depending on the purpose of their use, and the stage of development of the design.

After the Background section, we provide a summary of the requirements for a continuum between template and product description from various stakeholders' points of view in Section 4. We then explain our choice to use additional metamodels outside of the IFC standard, while still largely employing model transformation, model-to-text, and other model persistence formats. Finally the pragmatics of implementing import and export of product descriptions and their templates to and from various tools and file formats is explained, along with the role of modelling and transformations. We end with a discussion of future work.

2. BACKGROUND

2.1 IFC

Building Smart International is an industry consortium that defines and publishes standards for Building Information Model (BIM) interchange between computer applications used for architecture and engineering and construction. The centrepiece of their standards portfolio is the IFC specification, which is currently in its fourth major revision since first being published in 1995. The IFC specification is based around a metamodel written in the EXPRESS Schema language. EXPRESS Schemas support a similar expressive power to the combination of EMF and OCL.

IFC is a large specification containing over three-hundred type definitions and six-hundred classes. Classes are defined to represent all the major structural components of buildings, such as walls, doors, plumbing and electrical systems, air-conditioning, etc. These classes contain features which describe the common properties of building parts, such as their dimensions and subparts. In addition, there is a rich constructive solid geometry description used to represent shapes and relative positions of the structural elements can be related in space. Finally, a large range of classes which express relationships between building components is defined – 51 in all – covering things like physical containment, connectivity, spatial relations, grouping, nesting and sequencing. Classes are also defined for describing property sets associated with building components for more detailed and localised values to be defined about them. Most properties contained in these property sets consist only of named values with units of measurement.

Most classes representing building components may also have one or more shape (geometry) definitions, containing a set of linked geometric objects for use in a 3D viewer or CAD tool. However, the well-formedness rules of IFC require every file to start at the site level, and contain a building with at least one storey before a product element can be embedded into it, which makes it less than ideal for describing a simple thing like a window.

2.2 PSD

An additional Building Smart specification, Property Set Definitions (PSD) [4], provides an XML Schema for defining appropriate property sets for different contexts, without the need for the whole IFC Schema. Property sets may be defined for different trades in the building and construction industry, or different countries or other jurisdictions and information required by their building codes or other legislation. The PSD schema is relatively small, and allows for named sets of named properties to be defined, along with types for the values required and units of measurement. Everything is stored as strings, but with enough metadata to be able to convert them into numeric or enumerated types when used. Each property set definition specifies which IFC class types they may be relevant to. For example, a property set for use in thermal analysis in the Australian context may indicate that it is to be used to describe IfcWall and IfcWindow (and by implication all of their subtypes). In addition to their role as requirements specifications for required values to be provided for certain IFC classes, there is also an optional element in each property which allows for actual values to be inserted. There may be multiple values, enumeration values, as well as value ranges. The mapping from PSD to IFC's property set classes is not formally defined, and not fully isomorphic, but for most property types and values used in CAD tools, it is sufficiently straightforward.

2.3 BIM workbench in Eclipse

The basis for our implementation of IFC in the Eclipse Modelling Framework is a model-driven translation of the structural part of the IFC EXPRESS Schema into an Ecore metamodel [11] using a higher-order Tefkat transformation. The schema to metamodel translation is supported by a resource implementation for EXPRESS Part 21 files [7]which are used as the standard interchange format for IFC models by scores of CAD and CAE tools.

2.4 Repository as a Service

Our Web service based repository generation tool *Repository as a Service* (RaaS)[6, 5] takes an annotated Ecore metamodel as input, and creates WSDL and REST interfaces to an EMF Java model server. The web service interface allows remote (and local) model users to create, access and update medium-grained sub-graphs of an EMF model instance in a single operation invocation. The annotations of particular classes in the metamodel, chosen by the repository creator, makes these the access points to objects of that type, and all of the other objects linked to it by containment references (or other references chosen by the creator to relevant related model classes). The motivation for the development of RaaS is to provide distributed access to models specified by standard metamodels, such as UML, BPMN and IFC. Most current repositories operate on a file-based granularity, which has arbitrary contents at a very coarse grain, or via exposing individual programming language objects on a very fine-grained object or attribute per invocation basis. When a client of a model repository is also using EMF, the model is re-created transparently at the client side, but non-EMF clients can process models as XML documents or JSON structures. RaaS does allow us to directly access objects contained in an existing IFC file without needing to see the site/building/storey structure, but for import/export purposes, the well-formedness rules must be respected so that the files can be read by other tools.

3. OBJECT LIBRARIES

In the language used by Building Information Modellers (usually architects and engineers), the term *object library* is used to refer to a collection or repository of designs of building components, such as walls, air-conditioning units, furniture or any other part of a building model that can be packaged into units for re-use across different projects. Most CAD applications define their own file formats for storing these objects, such as Revit Families, or ArchiCAD GSM files. However, most also support an export mechanism to create IFC representations of these objects.

The new work in this paper is description of one of some of the implementation choices made when building on Object Library in partnership with the building agency of Queensland Government. All of the models and tools introduced in Section 2 are used to create the implementation. This Object Libray is aimed initially at describing the subset of building objects that are sold by manufacturers to the construction industry. Our rule of thumb is "Can you buy it at the hardware store?" These products are currently marketed to design agencies via printed catalogues (or internet-accessible PDF files) containing images and specifications of available products, sometimes including two dimensional plans and three dimensional renderings, as well as measurements and performance characteristics, in human-readable form. Occasionally CAD-tool-specific format object files are available to allow designers to incorporate a product into their building model, but most design practices create their own geometric model and properties, which they then store in a file system or CAD tool repository for use in future building designs. In large design practices, such as Project Services, where multiple CAD tools are in use, the designers often make separate objects for each tool representing the same design component. This is obviously more work, and also requires hand maintenance when objects are updated to reflect new products or design choices.

Our library aims to allow manufacturers to use tool-independent formats as the basis for properties and geometry of products, to be defined according to common national templates so that these can be discovered by keyword and/or property-based searching on the Web, and then transformed into the tool-specific format required. We also wish to download objects directly into the CAD tool, so that designers do not need to do file download management. In order to avoid the commoditsation of building products which have unique features, a standard set of properties will be used to describe every product of a certain category. Manufacturers may then add additional properties and information about their products, while still allowing the common characteristics of products of that category to be compared.

Ours is not the only tool-independent object library currently being developed. Other jurisdictions, including the USA, Norway and the Netherlands, are presently developing similar libraries. DDS have developed an early prototype illustrating the use of IFC as the basis for an object library [1]. The most advanced national project at the time of writing is the operational UK National BIM Library built by NBS [9], although it offers only CAD-tool-specific objects with varying coverage of formats for each object type available.

4. THE TEMPLATE / OBJECT CONTINUUM

In some design practices and projects, the actual building products to be used in construction of a building can be specified and have all their properties fixed at an early stage in the design. The architect may have experience with a particular manufacturer's product, and want to include it in the digital model right from the beginning. From this point of view, the product descriptions in the building product library are far too general, as they embody all of the options and variations of a product line, including finishes, colours, dimension variations, and other variable or multi-valued properties. The designer in this context will want to fix all the variables to include the fully specified product in the CAD design.

Normally in the design of most construction projects to be put out to tender, however, the design process may begin with very little detail about the objects in the digital design. Often the designer needs only to specify a few performance characteristics and perhaps identify some standards to which the products must conform, as required by local building codes and other laws and statutes. However, as the project moves towards the Request for Tender stage, a number of variables may be eliminated, and in particular, required dimensions will be known.

4.1 Level of Development

The concept of Level of Development (LOD), was adopted by the American Institute of Architects (AIA) in 2008. It was originally pioneered in the Vico software by the Webcor company, and known as Model Progression Specification (MPS). It was refined by the technology subcommittee of the AIA California Council's IPD Task Force, and was adopted by the AIA in late 2008. It is also known as the 'AIA E202'. Subsequently it has been widely adopted by the construction industry internationally as a convenient shorthand to define the steps through which objects used in BIM projects will progress as the design of a building or facility is developed and resolved.

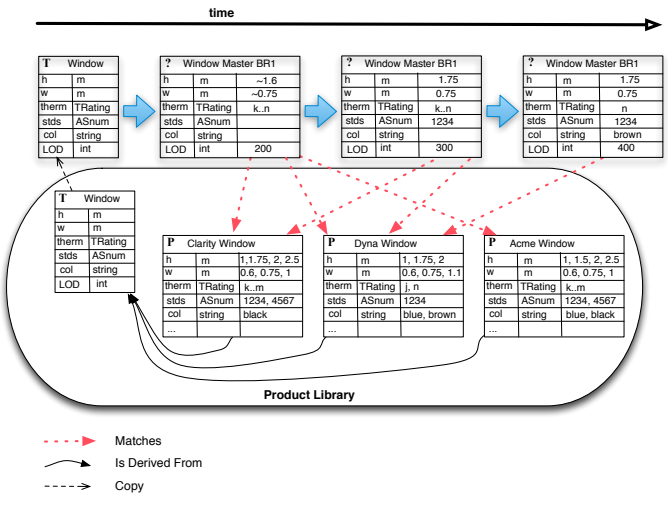

Figure 1: Relationships between Templates, Product Descriptions and Matching Design Requirements

From this point of view, the design starts out with an empty product template, in which the kind of object is known (door, window, beam, etc) and as the design evolves some properties of an object can be required to have values provided for them. Often LOD numbers are used to describe the stage at which the design is shared with other stakeholders in a project. The top row of "objects" in Figure 1 shows this accumulation of detail in the performance characteristics of an object over time. However, the choice of actual products to meet these criteria in some cases must be left to the contractor for legal or policy reasons, such as anti-corruption legislation, or cheapest supplier policy.

The way this issue is resolved in the building product library is that product templates and fully-specified CAD objects are two ends of a continuum, in which the product descriptions in the library occur part way. Templates will include property sets with the possibility to state that properties are of certain types (length, temperature, pressure) specified in particular units (milimetres, degrees celcius, bar) and must fall into particular ranges (200mm - 500mm) or have certain discrete values (200mm, 300mm, 400mm, 500mm). Each property will be either mandatory or optional when instantiated as a product description at a particular LOD.

Figure 1 shows a simplified view of the product library in the lower part of the diagram, with a set of windows conforming to the Window template (associated IFC geometry would also be available to preview and transform into tool-specific formats). Multiple values for properties indicate a product line with variations. The upper part of the diagram is an illustration of the use of more property values in a design object (usually inside a CAD tool) as the design matures over time.

4.2 Search using templates

Templates can be the basis for searches of the product descriptions in the building product library, in which case mandatory or optional characteristics for a property are replaced by an exact or approximate match, or are disregarded. For example, if a designer wishes to find all window systems that have a certain fire rating and which come in the approximate dimensions required for the design, she can select just three properties from a window template: Height, Width and Fire Rating, setting the others as "disregard". The first two would be set to "approximate match" with the height and width dimensions given, and the third to "exact match" with the fire rating required. Based on browsing of the matching windows the designer may choose some additional properties to be set in future searches to narrow down the range of matches. However, if no matches are found, then the dimensions may need to be re-examined.

The Matches arrows in Figure 1 show that the designer can use a partially populated set of properties to check that products are available to meet their increasingly detailed design requirements. Note that the "LOD" property increases over time, along with the number of properties containing values. Also, as properties become more specific, the set of matching products decreases. Designers may become aware that their requirements do not match any off-the-shelf products, and choose to commission bespoke building components, with the cost tradeoff that this implies.

The actual user interface to specify searches over templates in our demonstrator for an Australian National Object Library concept is much more fully featured than the diagram indicates, but space restrictions prevent its functionality being fully described here. In summary: properties can be uploaded to the library from existing design objects in CAD tools, and also downloaded back to the CAD tool to be attached to an existing CAD object from a selected product description (with or without geometry). Designers can choose which properties to match against, and how precisely, and the user is offered suggestions for search values based on the properties of products available. A product's definition may also be used as a search template to find similar products by choosing which properties to match, or adding additional values or ranges to the search.

5. PROPERTY SETS AS THE FUNDAMENTAL UNIT OF PRODUCT DESCRIPTION

As the previous section indicates, not all designs need to contain fully detailed product specifications, and in some cases this actually pre-empts the decisions that should properly be taken by others later in the design and construction process. It is commonly pointed out by evangelists for BIM in the construction sector that the difference between CAD and BIM is the 'I' in BIM, which stands for Information. In the context of the product description library, the information about a product is captured primarily in the property sets that are attached to the object, and the geometry is something that is useful to be able to preview, but could just as well come from a generic tool in a CAD product. The use of IFC as a metamodel and file format for products provides many benefits, including the ability to classify the kind of product according to its IFC classifier (IfcWall, IfcDoor, IfcFurniture, etc), and the set of standard attributes that is defined in order to describe an IFC object of that kind. Also, most CAD tools, although they may use different terminology to one another for describing building components, usually state which IFC class their own design elements correspond to, providing a level of correspondence, even without actually exporting a design element as an IFC file.

However, the number and scope of the inherent attributes of IFC objects are very limited. Often they include only bounding dimensions, and a reference to some material type from which a component is made. Most of the information in the IFC version of BIM lies in the property sets associated with an IFC object. These are usually populated by a CAD tool, or chosen on a jurisdiction, agency or even project-by-project basis, and vary widely between tools, countries and practices. Although the IFC metamodel contains a set of classes which describe properties and their types and units, and allow them to be grouped into property sets, the PSD XML schema represents much the same information, and it focuses only on representing the concepts relevant to property sets. Furthermore, IFC is known to most designers only as a file interchange format between tools, and they are not familiar with the attributes that are represented. Therefore, the graphical display of properties and their values that we wish to show a user of the product library must show both the 'inherent' IFC attributes, and the information available from the associated property sets in the same way (resembling the manner in which CAD tools reveal the properties

of a highlighted design object). In fact the only aspects of a specific product description that are specific to the IFC metamodel are: the name of classifier, which indicates the kind of product (IfcWindow, IfcFlowController, IfcColumn, etc) and its geometric shape description.

6. IMPLEMENTATION DETAIL

There are a number of emerging sources for the definition of object properties, based upon the IFC open standard. These are buildingSMART standard Property Set Definitions (PSDs), in the PSD XML format described earlier. Construction Operations Building information exchange (COBIE), and Specifiers' Properties Information Exchange (Spie) are US standards that are based on Excel spreadsheets. They have been developed to address differing industry requirements in a number of countries. They also address information requirements at different stages of the building lifecycle, and by a range of project participants. We chose the published COBIE spreadsheets as the basis for our initial object template imports, although we also experimented with available Spie and PSD definitions. However, inside the RaaS repository all these are stored in models based on the PSD XML Schema.

6.1 Implementation of Property Sets

In order to use the PSD XML Schema as an EMF model, we used the standard EMF Wizard to generate an Ecore Model from the Schema, which has the convenient side effect of also generating a resource implementation allowing the reading and writing of PSD-conformant XML documents to and from EMF model form. We defined a small model transformation to collect the attributes of each IFC metamodel class and creates an equivalent property set as a PSD model named <*IfcClassName*>Inherent. However, in the user interface we usually display this as "BaseQuantities", as the only properties usually contained are bounding dimensions.

The IFC metamodel is then used in the repository for storing only geometry descriptions. Property sets (including those derived from the IFC attributes) are stored as instances of the PSD metamodel. The values of properties can then be manipulated in one of two ways: firstly, through the retrieval via the Web service interface of RaaS as XML or JSON structures, or secondly through the resource implementation of PSD models as COBIE spreadsheets in the XLSX format. Via the web interface the product manufacturer can initiate an export of a COBIE spreadsheet file from a product description for download to their computer. The spreadsheet can then be populated with appropriate values, and re-uploaded via the user interface. The COBIE resource load implementation is in turn processed by a small jQVT transformation which maps each row in the spreadsheet to an object, and copies the contents of cells in the spreadsheet containing the values to its Value attribute. The transformation is declarative and easy to read, with the caveat that the class and attribute names in the Ecore generated from the PSD.xsd are somewhat obscure. However, jQVT generates Java code that performs these mappings inside the resource implementation in linear time.

6.2 Combining files

A product description is a folder containing a COBIE spreadsheet file describing the property sets, and optionally, an IFC file describing the geometry. But as we have observed, this geometry needs be be embedded into a structure describing a complete functional building, and always contains the nesting structure IfcProject/IfcSite/IfcBuilding/IfcBuildingStorey. In addition other elements need additional containing structure, for example a door (IfcDoor) needs to be hosted in a wall (IfcWall) which is located in a storey (IfcBuildingStorey). This approach facilitates the easy definition (for a CAD user) of the geometry for a product description in any CAD tool that supports IFC Export functionality - but resulting model contains a whole skeleton building design, which is not useful from a product definition point of view.

On the other hand property definitions are relatively minimal when expressed as COBIE spreadsheets, and one can edit them in Microsoft Excel or open source tools that understand the XSLX format. In addition the library's web interface also supports editing of the property sets by converting the spreadsheet to an EMF representation conforming to the PSD metamodel, and displaying the values as editable fields in the Web browser. From a technical point of view, this is done by providing an EMF resource implementation for the .xls file extension within the Eclipse framework. While using COBIE spreadsheets is the recommended way to get to product descriptions, we also allow for the import of the properties in existing IFC elements modelled in CAD tools, and ignore the extra containing structure. For most of the transformations, we use jQVT which supports the invocation of any method as well as evaluation of predicates. We use an open-source Java API to access and read a COBIE file, and jQVT then uses method calls as assertions in the target domain of a transformation relation.

6.3 A Summary of transformations used in the library implementation

There are several model transformation used in NOL: **COBIE to PSD** (xls2psd.jqvt): Used in the resource implementation of COBIE files; **PSD to COBIE** (psd2xls.jqvt): Used to propagate changes back from the PSD model to the spreadsheet; **IFC to PSD** (ifc2psd.jqvt): Used to derive "inherent" property sets from IFC metamodel classes; **IFC to GDL** (ifc2gdl.jqvt): Converts IFC geometry to GDL (the scripting language of ArchiCAD); and **psd_powerset.jqvt**: Originally used to group all PSD models provided by buildingSMART according to their IFC class types.

6.4 Limitations of Transformation

The transformation of geometry to Revit cannot be done by a conventional model transformation due to the proprietary nature of Revit file formats. Revit provides an API which has been used to develop a plugin which queries the object library using RaaS Web services calls. To facilitate this a .NET library of hand-written IFC classes has been developed, and IFC geometry object graphs are transfered to Revit using RaaS calls which transfer JSON messages and converted to Revit geometry in the plugin.

7. FUTURE WORK

There remain a number of unresolved issues which are the subject of future developments in the national object library. Chief among these is in clarifying the organisational and process requirements for the operation of the library, in terms

of defining who can and should be permitted/required to access or modify the product descriptions in the library, at the different stages of their lifecycles [3].

A second consideration is the need for large organisations, such as Project Services, to maintain separate object libraries for their own use, which refer back to national libraries and potentially add or refine properties of the product descriptions. In this stage many of the designs that will be stored in an object library will be systems composed of many different products and materials and will not be available "off the shelf" from any single manufacturer.This leads to a federation of libraries, and issues in terms of propagation/synchronisation of changes, and composition of systems of component objects from various sources.

This work uses the IFC classes as a primary classification into product categories. However, there are many national and international initiatives to classify building components and materials into categories by trade, by purpose, by stage of design and/or construction, and for many other purposes. The correct multiple classification of product descriptions according to several of these schemes is planned, and these are be likely to be supported in a semi-automated way through an ontology-based framework such as the Building Smart Data Dictionary [2].

Related projects based at QUT, with diverse partner organisations include: treating design specification documents as aspects of a BIM, and managing the synchronisation and consistency between them; cost planning using BIM with embedded objects from libraries; the transition of BIMs into Facilities Management platforms, including long-lived references to product information from object libraries as the basis of building maintenance, refurbishment and eventual destruction.

8. CONCLUSION

As an existing metamodel capable of capturing a great amount of detail about the geometry, structure and additional properties of buildings and their constituent objects, IFC, seems at first glance to be ideal for representing reusable design objects representing manufacturers' products in all their detail. However, upon closer inspection, even though we can apply a rich set of software tools to the creation and manipulation of such models, the incremental nature of the design process and the need to leave some parts of a design to other stakeholders means that completeness and detail are not always the primary considerations. Although we use a variety of models and serialisations, including IFC, which are not always elegant in their design or combination, an MDE approach allows us to use a combination of tools to reliably generate code and coordinate existing frameworks. The result is a more maintainable and adaptable object library with less hand-written code. The construction of the system in this way has also required us to explore ideas with wider applicability, such as the technological space integration, and the graduated approach to templates and objects, which could be usefully applied to other related physical modelling domains.

9. ACKNOWLEDGMENTS

This research was carried out at QUT as part of the activities of, and funded by, the Smart Services Cooperative Research Centre (CRC) through the Australian Government's CRC Programme (Department of Industry, Innovation, Science, Research & Tertiary Education). The Smart Services work is the technical part of the Object Libraries programme of the Sustainable Built Environment National Research Centre, which is a participant in and sponsor of the Smart Services CRC. We thank the Queensland Government's Project Services agency for its funding, engagement and leadership.

10. REFERENCES

[1] Bjorn K. Stangeland. http://www.dds-cad.net/files/net.dds-cad.com/downloads/Presseberichte/2011_IFC_for_Object_Libraries.P Accessed 26/7/2012.

[2] buildingSMART. http://www.ifd-library.org/. Accessed 26/7/2012.

[3] buildingSMART. Industry Foundation Classes, Edition 3, Technical Corrigendum 1. http://www.buildingsmart.com, July 2007.

[4] buildingSMART. IFC Property Set Definition XSD 2x3. http://www.buildingsmart-tech.org/psd/IFC2x3/final/index.htm, 2008. Accessed: 25/07/2012.

[5] K. Duddy. Building a USDL Repository as a Service. In A. Barros and D. Oberle, editors, *Handbook of Service Description – USDL and its methods*, pages 387–393. Springer, 2011.

[6] K. Duddy, M. Henderson, A. Metke-Jimenez, and J. Steel. Design of a model-generated repository as a service for USDL. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '10, pages 707–713, New York, NY, USA, 2010. ACM.

[7] International Standards Organisation (ISO). Industrial automation systems and integration – product data representation and exchange – part 21: Implementation methods: Clear text encoding of the exchange structure. ISO Standard 10303-21:2002, 2002.

[8] M. Lawley and J. Steel. Practical Declarative Model Transformation with Tefkat. In J.-M. Bruel, editor, *Satellite Events at the MoDELS 2005 Conference, Revised Selected Papers*, volume 3844 of *LNCS*, pages 139–150, Berlin, Germany, 2005. Springer Verlag.

[9] NBS. http://www.nationalbimlibrary.com. Accessed 26/7/2012.

[10] Object Management Group. MOF 2.0 Query/View/Transformation, version 1.1. OMG Document No. formal-2011-01-01, January 2011.

[11] J. Steel, K. Duddy, and R. Drogemuller. A Transformation Workbench for Building Information Models. In *Proceedings of the International Conference on Model Transformation*, 2011.

[12] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework, 2nd Edition*. Addison-Wesley Professional, 2nd edition, December 2008.